TP Sage.

Yannick Renard.

1. Introduction.

Le logiciel Software for Algebra and Geometry Experimentation (Sage) est un logiciel de mathématiques qui rassemble de nombreux programmes et bibliothèques libres sous une interface commune basée sur le langage de programmation Python. Il intègre des fonctionnalités utiles en cryptographie : la théorie des nombres, les corps finis, l'algèbre linéaire, les polynômes univariés et multivariés, ou les courbes elliptiques. Il bénéficie également d'extensions récentes (2009-2010) pour la cryptographie sur les S-Box et les fonctions booléennes. Ainsi, Sage a été utilisé, dès sa création, comme outil d'expérimentation au profit de la recherche en cryptographie.

Ce TP vous propose d'utiliser ce logiciel pour mettre en œuvre certaines notions utilisées en cryptographie symétrique comme l'interpolation de S-Box et de systèmes de chiffrement par blocs, ou l'application de l'algorithme de Berlekamp-Massey sur un chiffrement par flot.

Ce TP est organisé en deux grandes parties.

L'objectif de la première partie est de se familiariser avec les corps finis et les S-Box dans Sage, puis de représenter les S-Box comme des polynômes univariés à coefficients dans une extension de $\mathrm{GF}(2)$. On utilisera cette représentation pour appréhender l'interpolation d'un chiffrement par blocs basé sur un schéma de Feistel. On en déduira une méthode d'analyse polynomiale de S-Box.

L'objectif de la deuxième partie est de manipuler des registres à décalage à rétroaction linéaire (LFSR) et des fonctions booléennes, puis de construire un chiffrement par flot basé sur un LFSR filtré. On appliquera alors l'algorithme de Berlekamp-Massey sur les suites produites par un LFSR et un LFSR filtré, pour obtenir le polynôme de rétroaction du LFSR équivalent de taille minimale engendrant la même suite.

Le TP est construit de manière progressive, afin que vous puissiez appréhender une à une, chacune des notions précédemment évoquées. Pour définir des objets Sage ou appeler des méthodes Sage, vous pourrez vous référer au manuel d'utilisation de Sage joint en annexe et qui présente des exemples concrets. Des indications en italique sont là pour vous guider et donner les points clés.

2. Partie 1: Interpolation de chiffrement par blocs.

a. Corps fini.

L'objectif est de manipuler les objets corps finis dans Sage. On peut construire les corps composés $GF(q)$ avec $q = p^k$, p premier et $k > 1$ un entier. Un corps $GF(p^k)$ avec p premier et $k>1$ est isomorphe à l'anneau quotient des polynômes de $GF(p)[x]$ modulo un polynôme unitaire et irréductible de degré k. On utilise le constructeur GF, en précisant en paramètres le nombre d'éléments et le nom du générateur du corps.

Question 1 : définissez un corps fini K1 à 2^4 éléments.

Question 2 : quel polynôme irréductible est utilisé pour représenter ce corps ? Ce polynôme sera noté P1.

Utilisez la complétion automatique : taper K1. *suivi de la touche* Tab *permet d'afficher toutes les méthodes disponibles.*

Question 3 : définissez un anneau polynomial univarié sur $GF(2)$ en l'indéterminée X.

Question 4 : définissez alors le polynôme $P2=1+X^3+X^4$. Est-il irréductible ?

Utilisez la complétion automatique : taper P2. *suivi de la touche* Tab *permet d'afficher toutes les méthodes disponibles.*

Question 5 : utilisez ce polynôme pour définir un autre objet Sage K2 de type corps fini à 2^4 éléments. On obtient un corps isomorphe à K1.

b. Polynômes d'interpolation de S-Box.

L'objectif est de manipuler des objets S-Box dans Sage. Une boîte de substitution ou S-Box est un élément de base de la cryptographie symétrique. Elle prend m bits en entrée et renvoie n bits en sortie. Elle peut être vue comme une fonction booléenne vectorielle de $GF(2)^m$ vers $GF(2)^n$. La représentation entière est alors souvent utilisée, de sorte qu'une S-Box est décrite par sa table, c'est à dire 2^m entiers strictement inférieurs à 2^n.

Question 6 : définissez une S-Box de 4 bits, notée S, de table suivante (0,1,8,15,12,10,1,1,10,15,15,12,8,10,8,12)

On peut voir aussi cette S-Box de 4 bits comme une fonction polynomiale à coefficients dans $GF(2^4)$. Dans ce cas, ce polynôme est obtenu par la formule d'interpolation de Lagrange, qui est déjà implémentée dans le logiciel Sage. Il est appelé polynôme d'interpolation de la S-Box.

Question 7 : calculez son polynôme d'interpolation sur le corps fini à 2^4 éléments K1.

Utilisez la complétion automatique : taper S. *suivi de la touche* Tab *permet d'afficher toutes les méthodes disponibles.*

Pour obtenir la documentation d'une méthode, il suffit de taper S.nom_méthode ? *et d'évaluer la cellule de calcul.*

Question 8 : calculez son polynôme d'interpolation sur le corps fini à 2^4 éléments K2. Quelle conclusion peut-on en tirer ?

c. Interpolation d'un algorithme de chiffrement par bloc.

L'attaque par interpolation consiste à construire une fonction équivalente à tout ou partie de l'algorithme à partir d'un ensemble suffisamment grand de couples entrée/sortie. Elle peut s'appliquer à un élément particulier comme une S-Box ou à tout un algorithme. Quand elle est appliquée à l'algorithme de chiffrement ou de déchiffrement en entier, elle permet de chiffrer tout bloc clair ou de déchiffrer tout bloc chiffré sans connaître la clé secrète. Quand elle est appliquée à une partie d'un algorithme de chiffrement, elle permet de construire un distingueur capable de récupérer les sous-clés de ronde.

On considère ici un algorithme basé sur un schéma de Feistel et utilisant la S-Box étudiée précédemment. Il prend en entrée deux blocs de 4 bits notés xg et xd. Chaque bloc peut être vu comme une variable à valeur dans un corps fini à 2^4 éléments.

L'objectif est d'étudier l'interpolation de cet algorithme sur deux tours.

On utilise dans un premier temps la représentation de $GF(2^4)$ donnée par K1, défini précédemment. Dans K1, la S-Box est représentée par son polynôme d'interpolation I1.

Question 9 : définissez une fonction f1(x) qui renvoie I1(x).

Une fonction en Python se définit de la manière suivante :

```
def nom_fonction(paramètres séparés par des virgules sans précision de type) :
        return expression
```

Ecrivez directement le polynôme en le paramètre x *sans utiliser l'objet Sage* I1.

Question 10 : définissez un anneau polynomial multivarié sur K1 en xg, xd, k1 et k2, où k1 et k2 sont les clés des deux premiers tours.

Question 11 : exprimez les blocs de sortie gauche et droite yg2 et yd2 comme polynômes en les variables xg, xd, k1 et k2. Quels sont leur degré ?

Reportez-vous au schéma de Feistel joint en annexe. On ne se préoccupe pas de l'absence d'échange des blocs, spécifique au dernier tour.

On utilise maintenant la représentation de $GF(2^4)$ donnée par K2, défini précédemment. Dans K2, la S-Box est représentée par son polynôme d'interpolation I2.

Question 12 : définissez une fonction $f2(x)$ qui renvoie $I2(x)$.

Question 13 : définissez un anneau polynomial multivarié sur $K2$ en xg, xd, $k1$ et $k2$, où $k1$ et $k2$ sont les clés des deux premiers tours.

Question 14 : exprimez les blocs de sortie gauche et droite $yg2$ et $yd2$ comme polynômes en les variables xg, xd, $k1$ et $k2$. Quels sont leur degré ?

La complexité en temps de l'interpolation de Lagrange d'un polynôme de degré k-1 est en $\theta(k^2)$. La quantité de données nécessaires est k couples entrée/sortie.

Question 15 : quelle représentation du corps à 2^4 éléments un attaquant a intérêt à utiliser, s'il veut interpoler cet algorithme ? En déduire une méthode pour tester la résistance d'une S-Box à l'interpolation.

d. Analyse polynomiale d'une S-Box.

L'objectif est de mettre en pratique ce qui a été vu jusqu'à présent pour évaluer un critère de sécurité de S-Box face à l'interpolation.

Question 16 : définissez une fonction qui retourne la liste de tous les polynômes irréductibles à coefficients dans $GF(2)$ de degré n.

Utilisez la méthode polynomials *de l'objet anneau polynomial univarié sur* GF(2)*, en lui passant le paramètre* of_degree=n*. Pour obtenir la documentation de la méthode, il suffit de créer l'anneau polynomial* R*, de taper* R.nom_méthode ? *et d'évaluer la cellule de calcul.*

Une liste se définit par les symboles [ ] *sans préciser le type des éléments. On peut créer une liste vide puis utiliser la méthode* append *pour ajouter des éléments. Le langage Python permet aussi de transformer des listes très facilement :* [expression(i) for i in liste if condition].

On étudie une première S-Box de 8 bits définie par la table suivante :

(22,71,180,229,241,160,83,2,123,42,217,136,156,205,62,111,204,157,110,63,43,122,137,216,161,240,3,82,70,23,228,181,1,80,163,242,230,183,68,21,108,61,206,159,139,218,41,120,219,138,121,40,60,109,158,207,182,231,20,69,81,0,243,162,56,105,154,203,223,142,125,44,85,4,247,166,178,227,16,65,226,179,64,17,5,84,167,246,143,222,45,124,104,57,202,155,47,126,141,220,200,153,106,59,66,19,224,177,165,244,7,86,245,164,87,6,18,67,176,225,152,201,58,107,127,46,221,140,74,27,232,185,173,252,15,94,39,118,133,212,192,145,98,51,144,193,50,99,119,38,213,132,253,172,95,14,26,75,184,233,93,12,255,174,186,235,24,73,48,97,146,195,215,134,117,36,135,214,37,116,96,49,194,147,234,187,72,25,13,92,175,254,100,53,198,151,131,210,33,112,9,88,171,250,238,191,76,29,190,239,28,77,89,8,251,170,211,130,113,32,52,101,150,199,115,34,209,128,148,197,54,103,30,79,188,237,249,168,91,10,169,248,11,90,78,31,236,189,196,149,102,55,35,114,129,208)

Question 17 : calculez tous les polynômes d'interpolation de cette S-Box pour toutes les représentations possibles de $\mathrm{GF}(256)$. Affichez le degré de ces polynômes. La S-Box est-elle robuste à l'interpolation ?

On étudie une deuxième S-Box de 8 bits, celle de l'AES définie par la fonction inverse à transformation linéaire inversible près.

Question 18 : calculez tous les polynômes d'interpolation de cette S-Box pour toutes les représentations possibles de $\mathrm{GF}(256)$. Affichez le degré de ces polynômes. La S-Box est-elle robuste à l'interpolation ?

L'AES est déjà implémenté dans Sage. On peut ainsi récupérer très facilement sa S-Box de la manière suivante : $\mathrm{mq.SR}(10,4,4,8).\mathrm{sbox}()$.

3. Partie 2 : LFSR filtré.

e. LFSR

Un registre à décalage à rétroaction linéaire ou Linear Feedback Shift Register (LFSR) permet d'engendrer une suite pseudo-aléatoire, vérifiant une relation de récurrence linéaire. Un LFSR de n bits peut être représenté par son polynôme de rétroaction de degré n dont les coefficients définissent la relation de récurrence. Du fait des propriétés statistiques obtenues et de l'optimalité en terme de période et de complexité linéaire (taille du plus petit LFSR équivalent), les polynômes de rétroaction utilisés sont primitifs.

Question 19 : définissez une fonction qui calcule les n bits de sortie d'un LFSR à partir de l'expression de son polynôme de rétroaction et de son état initial.

Utilisez le code de la fonction présenté dans le manuel Sage joint en annexe.

Question 20 : soit le polynôme $P=1+X^3+X^4$. Est-il primitif ?

Un anneau polynomial univarié sur $GF(2)$ *en l'indéterminée* X *a déjà été défini dans la première partie.*

Utilisez la complétion automatique : taper P. *suivi de la touche* Tab *permet d'afficher toutes les méthodes disponibles.*

Question 21 : calculez la suite L de longueur 15 produite par le LFSR défini par le polynôme de rétroaction P et l'état initial $(s0,s1,s2,s3)=(1,0,1,1)$.

L'algorithme de Berlekamp-Massey donne le polynôme de rétroaction du LFSR de taille minimale engendrant une suite. Le LFSR est unique si et seulement si la longueur de la suite est double de la taille du LFSR.

Question 22 : utilisez l'algorithme de Berlekamp-Massey sur la suite L. Vérifier qu'on retrouve bien le polynôme de rétroaction P.

f. LFSR filtré

On s'intéresse aux algorithmes de chiffrement par flot à transition linéaire. La fonction de transition linéaire est généralement construite avec un ou plusieurs registres à décalage à rétroaction linéaire (LFSR). La fonction de filtrage non linéaire garantit que la suite chiffrante produite ne dépend pas linéairement de l'état initial du ou des LFSR. Elle est construite avec une fonction booléenne prenant en entrée, soit les bits de sortie des LFSR dans le cas de LFSR combinés, soit certains bits de l'état interne d'un unique LFSR dans le cas d'un LFSR filtré. Pour la suite, on considère le cas d'un chiffrement par flot à transition linéaire basé sur un LFSR filtré.

Question 23 : définissez un anneau polynomial booléen en les variables x0, x1, x2, x3. Définissez ensuite une fonction booléenne f de forme algébrique normale x0+x1*x2+x0*x2*x3.

Utilisez le constructeur BooleanPolynomialRing*. Reportez-vous au manuel Sage.*

On utilise cette fonction booléenne pour filtrer un LFSR. Cette fonction prend en entrée les bits 0, 1, 2 et 5 de l'état interne du LFSR et renvoie un bit de suite chiffrante.

Question 24 : définissez une fonction qui calcule les n bits de sortie d'un LFSR filtré par la fonction booléenne f précédente, à partir de l'expression de son polynôme de rétroaction et de son état initial.

Transformez la fonction implémentant un LFSR en modifiant les valeurs renvoyées.

Un LFSR filtré est en fait équivalent à un LFSR simple de grande taille. Il existe une borne théorique de la complexité linéaire (taille minimale) du LFSR simple équivalent au LFSR filtré (de taille n, filtré par une fonction booléenne f) : $\sum_{0 \leq i \leq deg(f)} C(n,i)$.

On va vérifier cette borne dans le cas du LFSR de polynôme de rétroaction $1+X+X^6$, d'état initial [1,0,1,1,0,0], filtré par la fonction booléenne f vue précédemment.

Question 25 : exprimez la borne de la complexité linéaire, notée CL.

Utilisez la fonction Sage binomial *et la transformation de liste très utilisée en Python :* [expression(i) for i in liste if condition] *pour créer la liste des coefficients binomiaux. La somme des éléments d'une liste se fait alors simplement en passant cette liste en paramètre à la fonction* sum.

Question 26 : calculez la suite L1 de longueur 2*CL produite par le LFSR défini par le polynôme de rétroaction $1+X+X^6$ et l'état initial (s0,s1,s2,s3)=(1011), filtré par la fonction booléenne f.

Question 27 : utilisez l'algorithme de Berlekamp-Massey sur la suite L1 pour obtenir le polynôme de rétroaction P-EQUIV du LFSR simple équivalent. Quel est son degré d ?

Question 28 : définissez une liste, notée INIT, des d premiers éléments de la suite L1 produite par le LFSR filtré. Cette liste est l'initialisation du LFSR simple équivalent.

Question 29 : calculez la suite L2 de longueur 2*CL produite par le LFSR simple de polynôme de rétroaction P-EQUIV trouvé à la question précédente et d'état initial INIT. Vérifier qu'on retrouve bien la suite L1.

Utilisez ici la fonction implémentant un LFSR et pas celle implémentant le LFSR filtré.